**TCP Protokol**

*Profesor*: Branislav Atlagić

*Predmet*: Industrijski komunikacioni protokoli u elektroenergetskim sistemima

*Tema*: TCP Protokol

*Autori*:

1. Marko Maleš PR62-2016
2. Stela Dolinaj PR56-2016

***Uvod***

Projekat predstavlja pojednostavljenu implementaciju TCP protokola. Početna tačka projekta je bila komunikacija zasnovana na UDP protokolu. Autori su imali zadatak da tu komunikaciju dopune sa klasičnim funkcionalnostima TCP protokola. Te funkcionalnosti su sledeće: sliding window, buffer pool, timeout, CRC provera i ACK funkcionalnost. Način na koji su funkcionalnosti TCP protokola implementirane biće ovde izložen. Takođe, u zaključku će biti prezentovane dodatne funkcionalnosti kojima bi se ovaj projekat mogao poboljšati, a koje su radi jednostavnosti izostavljene.

***Opis***

Projekat je zasnovan na dvosmernoj komunikaciji Klijent-Server. Tačno jedan klijent učestvuje u komunikaciji sa serverom.

Aplikacija omogućava klijentu da šalje poruke serveru u obliku segmenata (delova poruke). Klijent za svaki poslati segment čeka povratnu poruku od servera kojom server klijentu potvrđuje da je server primio određeni segment. Povratna poruka sadrži jedinstveni identifikator segmenta koji se potvrđuje.

Nasuprot tome, server prima segmente i šalje klijentu potvrdu za svaki segment poruke koji je primljen, ukoliko je korektan. Kriterijum za korektnost segmenta predstavlja prolazak CRC provere.

Dodatno, klijent obavlja proveru kašnjenja potvrde segmenta. Ukoliko vreme predviđeno za primanje potvrdne poruke od servera istekne pre primanja potvrde na klijentu, klijent ponavlja slanje svih segmenata počev od onog segmenta za koji se desilo kašnjenje.

Podaci za razmenu se smeštaju u posebnu strukturu za prihvat podataka - buffer pool.

***Upotreba niti u projektu***

Budući da je projekat ekstenzivan i pogoduje paralelizaciji, kod na klijentu i serveru je podeljen na niti. Rad na klijentu je podeljen između dve niti. Prva nit služi za segmentiranje poruka i slanje segmenata serveru kao i računanje CRC vrednosti za svaki segment, dok druga nit ima za posao da prima poruku o uspešnom primanju segmenata na serveru i proverava da li je istekao predviđeni vremenski period za primanje takve poruke za svaki pojedinačni segment. Ove dve niti komuniciraju uz pomoć promenljivih, a štetno preplitanje niti je preventirano uz pomoć upotrebe kritičnih sekcija. S druge strane, na serveru je rad takođe podeljen na dve niti. Prva nit je zadužena za primanje segmenata od klijenta, smeštanje segmenata u strukturu za prihvat podataka, kao i računanje CRC provere za svaki pristigli segment. Druga nit služi za slanje povratne poruke klijentu u kojoj se navodi identifikator sekvecijalno poslednjeg uspešno pristiglog segmenta. Sekvencijalno poslednji uspešno pristigli segment je onaj segmnet za koji važi da je svaki segment pre tog takođe stigao i prošao CRC proveru.

Upotrebom niti, projekat je pojednostavljen svojom modularnošću i ubrzan uz pomoć paralelizacije rada.

***Funkcionalnosti***

Od autora se tražilo da implementiraju sledećih pet funkcionalnosti.

*1.Sliding window*

Klijent serveru šalje poruke u obliku segmenata odnosno delova poruke. Klijent poruku deli na segmente dužine 64KB (kilobajta). Za svaki segment koji klijent pošalje serveru, klijent očekuje da dobije povratnu poruku o uspešnom primanju tog segmenta na serveru. Ta potvrdna poruka će se u daljem tekstu zvati ACK (eng. acknowledge). ACK za svaki segment mora stići onim redosledom kojim su segmenti slani na klijentu. Klijent ima određeni vremenski period koliko sme da čeka na potvrdnu poruku za određeni segment. Taj vremenski period će se u daljem tekstu zvati TIMEOUT i on predstavlja vreme izraženo u milisekundama. Za svaki segment koji pošalje serveru, klijent pamti trenutak u kom je segment poslat izražen u milisekundama. U toku svog rada, klijent periodično propituje razliku između trenutnog vremena i trenutka kada je određeni segment poslat. Propituju se samo segmenti koji su poslati, ali za koje nije stigao ACK. Ukoliko dođe do situacije da je više od TIMEOUT milisekundi prošlo od slanja nekog segmenta, može se reći da je došlo do kašnjenja. U tom slučaju, klijent ponavlja slanje svih segmenata počev od segementa za koji se desilo kašnjenje. Ova funkcionalnost gde klijent ponavlja slanje segmenata kada ACK izostane u predviđenom vremenskom periodu TIMEOUT, naziva se *sliding window*.

*2.Buffer pool*

Funkcionalnost *buffer pool* se odnosi na posebnu strukturu podataka koju klijent i server koriste za skladištenje i prihvat segmenata, respektivno. Buffer pool struktura u sebi sadrži pokazivače na delove memorije koji se koriste za skladištenje segmenata. Delovi memorije za skladištenje segmenata će se na dalje zvati *baferi*. Takođe, sadrži i naznaku koji od tih bafera je trenutno u upotrebi. Značenje te naznake je drugačije na klijentu i serveru. Na klijentu, naznaka označava da li je u određenom baferu smešten segment za koji još nije stigao ACK. U tom slučaju, bafer ne može ponovo da se koristi, jer se segment u tom baferu mora sačuvati za slučaj potrebe za ponovnim slanjem tog segmenta. Inače se bafer oslobađa tako što se vrednost naznake menja i bafer se može ponovo koristiti. Na serveru, naznaka označava da li je segment iz određenog bafera pročitan od strane servera. Bafer se smatra pročitanim kad se za njega izvrše neophodne provere tačnosti segmenta u tom baferu i pošalje povratna poruka tj. ACK klijentu. Ukoliko ACK za segment u određenom baferu nije poslat, bafer se ne može koristiti za prihvat novopristiglih segmenata. U suprotnom, buffer se oslobađa postavljanjem nove vrednosti naznake za taj bafer i može se ponovo koristiti.

*3.Timeout*

Funkcionalnost *timeout* se odnosi na ograničenost vremenskog perioda između trenutka kada klijent šalje segment i trenutka kada klijent prima ACK za taj isti segment. Vremenski period između ta dva trenutka nazvaćemo PERIOD i izražavaćemo ga u milisekundama. PERIOD poseduje gornju granicu koju koju ne sme preći nazvanu TIMEOUT. TIMEOUT je izabran da bude 100 milisekundi. Izbor o dužini TIMEOUT-a je napravljen sa idejom da TIMEOUT bude kraći od ukupnog vremena potrebnog da segment stigne od klijenta do servera i da od servera do klijenta stigne ACK za taj segment. PERIOD se periodično izračunava za svaki pojedinačni segment. Slučaj kada je vrednost PERIOD-a duža od TIMEOUT-a za određeni segment naziva se kašnjenje. Ukoliko dođe do kašnjenja za neki segment, klijent ponavlja slanje svih segmenata počev od tog segmenat. Ova funkcionalnost se implementira samo na klijentu.

*4.CRC provera*

CRC (eng. Cyclic Redundancy Check) je način da se proveri da li je došlo do nenamerne izmene podataka prilikom slanja. CRC se implementira i na klijentu i na serveru, ali na različite načine. CRC provera predstavlja algoritam uz pomoć kog klijent računa jedinstvenu vrednost za određenu poruku koju šalje. Ta jedinstvena vrednost predstavlja ostatak pri deljenju poruke sa polinomom *x^7+x^6+x^4 + x^3* odnosno binarnom predstavom tog polinoma *b11011000*. Stepeni polinoma su izabrani nasumično. Deljenje se obavlja tako što se od najvišeg ka najnižem bitu poruke uzimaju delovi poruke veličine 1B (bajt). Zatim se svaki takav deo poruke deli sa polinomom odnosno XOR-uje sa polinomovom binarnom predstavom i rezultat se smešta u poruku na mesto tog dela poruke. Ovakvo deljenje se obavlja dok se ne dođe do poslednjeg bajta poruke. Za poslednji bajt poruke se ne računa količnik sa polinomom, već se on smatra ostatkom pri deljenju poruke sa polinomom. Taj ostatak klijent dodaje na kraj poruke i zajedno ih šalje serveru. Dodavanjem ostatka na kraj poruke, čini celu poruku deljivom sa tim polinomo. Server prima poruku i za nju računa ostatak pri deljenju sa istim tim polinomom. Server zatim proverava da li je ta vrednost jednaka nuli. Ukoliko jeste, poruka je prošla CRC proveru, jer je poruka deljiva sa polinomom. Ukoliko nije, poruka se smatra korumpiranom prilikom slanja i stoga server neće slati potvrdu za tu poruku klijentu. Loša strana CRC provere je traženje pogodnog polinoma za određenu primenu.

*5.ACK*

ACK predstavlja povratnu poruku od servera ka klijentu. Server njom potvrđuje da je primio segment od klijenta i da je taj segment prošao CRC proveru. ACK se sastoji od jedinstvenog identifikatora segmenta koji se potvrđuje (u našem projektu to je indeks segmenta u poruci).

***Zaključak***

Projekat predstavlja ozbiljno pojednostavljenje TCP protokola, koliko se i može očekivati od rada ovog obima. Ipak, moguća poboljšanja se mogu uočiti. Projekat bi se znatno konkretizovao radom sa više klijenata. Zatim, iako je napravljen trud da se paralelizuje komunikacija ka i od servera na klijentu, poruke se i dalje šalju sekvencijalno što predstavlja usporenje izvršavanja programa. I na kraju, testiranje aplikacije je izostalo, što je potencijalno moglo otkriti greške u radu programa, kao i optimalna ograničenja dužine segmenata koji se šalju ka serveru. Uprkos tome, aplikacija predstavlja skladnu celinu koja obavlja jednostavnu komunikaciju Klijent-Server uz pomoć TCP protokola, što je od autora i traženo.